2018 30th International Symposium on Computer Architecture and High Performance Computing

MLNoC: A Machine Learning based approach to NoC design

Nishant Rao Akshay Ramachandran Amish Shah
NetSpeed Systems
{nishant, akshay, amish}@netspeedsystems.com

***Abstract*—Modern System on Chips (SoCs) are becoming increasingly complex with a growing number of CPUs, caches, accelerators, memory and I/O subsystems. For such designs, a packet based distributed networks-on-chip (NoCs) interconnect can provide scalability, performance and efficiency. However, the design of such a NoC involves optimizing a large number of variables such as topology, routing choices, arbitration and quality of service (QoS) policies, buffer sizes, and deadlock avoidance policies. Widely varying die sizes, power, floorplan and performance constraints across a variety of different market segments, ranging from high-end servers to low-end IoT devices, impose additional design challenges.**

In this paper we demonstrate that there is a strong correlation between SoC characteristics and good NoC design practices. However this correlation is highly non-linear and multi-dimensional, with dimensions indicative of the features of the SoC, design goals and properties of the NoC. This results in a high-dimensional NoC design space and complex search process which is inefficient to solve with classic algorithms. Using a variety of real SoCs and training data sets, we demonstrate that a machine learning (ML) based approach yields near-optimal NoC designs quickly. We determine a number of SoC and NoC features, describe reduction methods, and also show that a multi-model approach yields better designs. We demonstrate that for a wide variety of SoCs, ML based NoC designs are far superior to those designed and optimized manually over years on almost all quality metrics.

Keywords-Network-on-chip, Machine learning, Interconnect, System-on-Chip, Architecture

I. INTRODUCTION

System on Chips (SoCs) are rapidly evolving, becoming increasingly sophisticated, feature rich, and high performance by integrating a growing number of CPU cores, memory and I/O subsystems, and specialized acceleration IPs. Today, higher performance is not only achieved by adding more CPU cores. Instead, heterogeneous system architectures are gaining popularity where specialized accelerators are used to offload common operations. Such designs offer unprecedented performance and efficiency due to the tight integration of CPUs, GPUs and custom hardware accelerators with the system I/O bus and memory. This movement along with the need for high performance and power efficient shared memory communication has driven the need for sophisticated on-chip interconnect (NoC) solutions. Interconnects play a crucial role in achieving desired system performance and power efficiency. In the past, architects have designed interconnects relying on their experience and making key design decisions such as choice of topology and routing based on intuition and prior experience. However, this approach does not scale for heterogeneous systems where on-chip requirements and design constraints are extremely diverse. Due to the complexity of interactions among various components, it is practically impossible to design a near optimal, functionally correct, high performance, and low power interconnect that satisfies all use cases.

To address this, automated NoC design flows are gaining popularity, where software algorithms are used for designing and analyzing NoC architectures based on high-level system requirements. Varying degrees of automation have been introduced from partial topology generation and bandwidth analysis to formal method based fully automated construction that guarantees an optimal solution[26][27]. The approach generally begins with capturing the system requirements such as traffic bandwidth, latency and power constraints. Based on these, several NoC topology choices are explored, analyzed through static analysis and dynamic simulation driven methods and selected based on user provided quality metrics. Formal methods are also deployed to ensure that each design satisfies the user constraints and is functionally correct in terms of being deadlock-free and Quality of Service (QoS) aware. For small SoCs, the entire design space can be explored and a near-optimal solution can be obtained reasonably quickly. However, the design space for large chips is unreasonably vast and highly non-linear, making it extremely challenging to either reduce the design space effectively or quickly scan and converge on an optimal solution.

This paper introduces MLNoC, a solution that utilizes machine learning to quickly explore and evaluate a multitude of possible NoC solutions to determine an optimal interconnect design. MLNoC uses extensive training data based on a large number of real world and synthetic designs to learn which design approach works well for different kinds of requirements and constraints. The pre-trained ML model used in MLNoC enables it to quickly steer towards good design points avoiding large regions in the design space altogether enabling significantly faster convergence for a new system requirement and constraint. The training process of MLNoC models involves identifying the key features and constraints of the system, using various NoC design approaches and evaluating the resulting NoC quantitatively. The fully trained model is then used to intelligently map the functional and design goals of the SoC to a smartly reduced subset of the design techniques, resulting in near-optimal power, performance and area. Since different SoC designs may have different optimization goals such as low power, or

978-1-5386-7769-8/18/$31.00 ©2018 IEEE
DOI 10.1109/SBAC-PAD.2018.00015

high performance or reduced die size, MLNoC uses multiple models each trained towards optimizing each design goal. These models are used in collaboration to obtain the best design for any user defined quality metric.

In our experiments based on 30 real-world SoC designs spanning different market segments, we find that MLNoC consistently produces NoCs with superior performance, power and area than a hand optimized design. On average, MLNoC achieve about 50% higher bandwidth and have 11% lower area than the NoCs designed by expert NoC architects over several weeks. Since different SoC designs have different design goals, we train multiple ML models each geared towards maximizing different quality metrics. We show that for every SoC design and for every quality metric the corresponding ML model produces better results. We also show that quality of results produced by a ML model is highly dependent on the size and quality of the training data. By progressively increasing the amount of training data to the ML models, the average quality of results produced by MLNoC increases by more than 125%. The remainder of the paper describes our approach in greater detail, followed by comprehensive experimental results.

II. Related Work

Our proposed solution to using machine learning for NoC design touches on many problems that have been proposed in the literature, such as NoC micro-architecture, power management schemes, QoS in networks-on-chip as well as different machine learning techniques. In this section, we review some of this related work.

The early focus on NoC construction involved reducing wiring and improving flexibility by moving to packet based switching instead of the traditional circuit switched network [1]. Different types of NoC architectures have been explored in prior research. Reference [2] proposed designing a full mesh NoC, with a switching element at every cell that is connected to four other neighboring switches. Bus-based NoC architectures were explored and analyzed in [5]. NoC topologies moved towards tree-based designs with the aim of minimizing network delay and energy consumption in [6]. The foundations of a layered micro-network based NoC for SoC connectivity was proposed and explored in [3], where the authors propose that creating complex SoCs requires a modular, component-based approach to both hardware and software design. New methodologies to manage power consumption in NoCs in a node-centric manner where local power managers are used for managing power and QoS was suggested in [4].

In the machine learning realm, Decision-tree based classifiers have been attractive owing to their low execution times, but they cannot be grown to accommodate high levels of complexities owing to possible loss in accuracy. In an effort to overcome this problem, [7][8][9] proposed a method of building a cluster of decision trees which work on subsets of the feature space, and empirically proved that the accuracy improved. Some basic analysis on comparisons of different types of classifiers such as Nearest Neighbor Classifier, Tangent Distance Classifier and Optimal Margin Classifier was shared in [10]. New ways of combining multiple classifiers using neural networks, termed Combination of Multiple Classifiers (CME) was studied in [11]. Studies related to Neural-network-based fuzzy classifiers, that cover aspects of fuzzy rule generation, handling fuzzy input and providing fuzzy classification have been covered in [12][13][14].

III. Current Challenges in NoC Design

A. Complex Multi-variable Problem

At the heart of any modern SoC is a sophisticated interconnect that provides on-chip connectivity and key system services such as cache coherency, quality of service (QoS), power management and functional safety. For any combination of requirements, there is a multitude of design possibilities for an architect to choose from, including topology, routing mechanisms [15][16][17][18][19][20], virtual channels, routes, QoS algorithm [28], arbitration policies, deadlock avoidance rules, channel widths heterogeneity for performance, buffer sizes, clock domains, and power and voltage domains. Each combination of design strategies may lead to a different interconnect with unique performance and cost etc.

B. Large Design Space

Identifying the right combination of design strategies that lead to near optimal interconnect design for any given system specification is challenging. This is because the perfect strategy depends upon a large number of SoC parameters. These include floorplan, routing constraints, resources available, connectivity requirements, protocol level dependency, clock characteristics, process characteristics such as wire delay, power budgets, bandwidth and latency constraints, etc. As a result, the number of unique dimensions in the design strategies space grows to several hundred causing excessively large design space. Additionally, several strategies are interdependent on others, which makes it difficult to pick each strategy in isolation from the others.

C. Comfort around Simplistic Designs

To address complexity, SoC architects tend to rely on simple rules of thumb for interconnect design. For example, they may decide to use a mesh topology for high all-to-all bandwidth but a tree topology for maximizing bandwidth around a few memory controllers, a ring for simple routing, and star for simple crossbar design. Another motivation behind using such designs is that it is easier to ensure architecture correctness and analyze system performance. Fig. 1 illustrates a few standard NoC topologies.

D. Performance, Power and Area Tradeoffs

Performance, Power and Area (PPA) are the three golden metrics in chip design. Modern chips designed to target very high bandwidths require NoCs that minimize bottlenecks in the design. With chips aiming to become smaller in size with every passing year, the area of silicon is a premium. In addition to minimizing wiring and number of flops while meeting high bandwidths, all chips have limits on the power

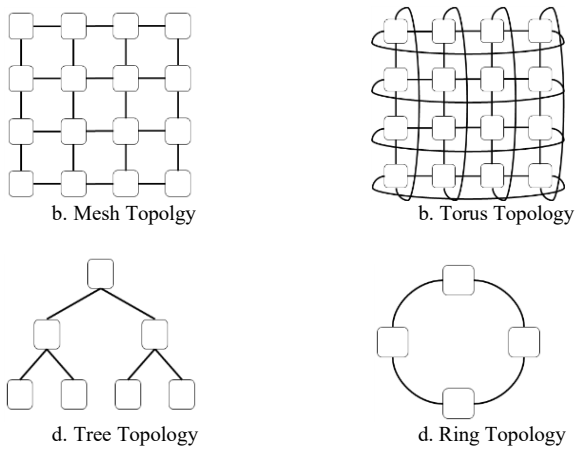

Fig. 1. NoC Topologies

consumption and efficient power management schemes [22][23][24][25] such as clock gating are an important aspect of interconnect construction. The requirements of PPA are varied depending on the use case of the applications - Data center chips often try and achieve very high bandwidths like 5 Tbps while mobile chips have very strict area and power constraints. Due to these tradeoffs, a one-size-fits-all approach to NoC design does not yield the most optimal results.

E. Buffer Sizing and Pipelines

Modern chips have strict design constraints on area requirements, and much of the area is constituted by wires, flops and RAMs. Buffer sizing is an important aspect of interconnect construction [21], and improper buffer sizing could lead to over provisioning of resources on one hand or hurt performance on the other. Without intelligent sizing, the cost of buffers could be significant in terms of performance or area. Pipelines need to be inserted in wires in order to ensure that signals can successfully traverse from one part of the wire to the other within the time constraint associated with the frequency of operation.

IV. Machine Learning Based NoC Construction

In the high-dimension space of NoC design, near-optimal design points are generally relatively rare and their regions can vary widely based upon the SoC requirements. It is impractical to search the entire design space. It is also impractical to devise a polynomial time algorithm due to the highly nonlinear nature of the space. For such problems, machine learning (ML)—where past experiences in solving similar problems are used in the form of training data to learn, build predictive models and find solutions for new similar problems—shows tremendous promise.

Machine learning problems involve defining a set of inputs and outputs that have a typically complex relationship that is scored based on a defined quality metric. In order to apply machine learning to the problem of NoC design, we use the SoC specification as an input and the design strategies used to construct the NoC as an output with a quality that is based on the design goals of the SoC.

A. SoC Features

In order to make use of machine learning to choose a NoC architecture for a given SoC, it is necessary to extract the features of the SoC specification that affect the interconnect design. These features are converted into one or more bit vectors that serve as an input to the ML process.

1) Feature Extraction

We classify the features of the SoC into three categories viz. grid features, traffic features and topology features that all strongly impact the NoC design and are largely non-overlapping with respect to one another.

a) Grid Features

The parts of the SoC specification that affect the NoC backbone architecture are classified under grid features. This includes the number of agents, number of ports that connect to the NoC and their protocols, the widths of different NoC interfaces, the clock frequency, power and voltage partitioning in the SoC, the presence of coherent ports, the address widths and system address maps. These features may influence the NoC topology, protocol used in the NoC, clock and power management schemes, buffer sizes, etc.

b) Traffic Features

The traffic features include parts of the SoC specification that influence the connectivity provided by the NoC. This includes the point-to-point connectivity between agents, types of interfaces/channels used for communication such as reads, writes, snoops, dependencies between different interfaces in the NoC, bandwidth and latency requirements for each agent, QoS and real-time requirements. These features may influence the span of the NoC within the SoC, the use of different routing physical channels, virtual buffers/channels to prevent network-level deadlocks and ensure performance, width of NoC channels, depth of NoC buffers and arbitration schemes.

c) Topological Features

The topological features include the size of the SoC, size and positions of agents, space available for NoC channel routing, wire density constraints, flop density constraints and process node technology. Topological features influence the NoC topology, routing choices between agents, use of physical and virtual channels, arbitration and number of pipelines on wires.

2) Feature Reduction

The features described above represent only a handful of the SoC characteristics that affect NoC design. In reality, there are thousands of SoC features, some that affect the design of interconnects and others that do not. Selecting the important features that affect the NoC design is a critically important step in crafting an efficient predictor. Having too many features results in a large input vector to the predictor which not only increases the dimension and search space but can also lead to noise in the data. Using various dimension reduction techniques, these features are reduced and grouped into the grid, traffic and topology categories. It is also important to keep features linear and minimize overlaps between them. Fig. 2 illustrates an entropy analysis between

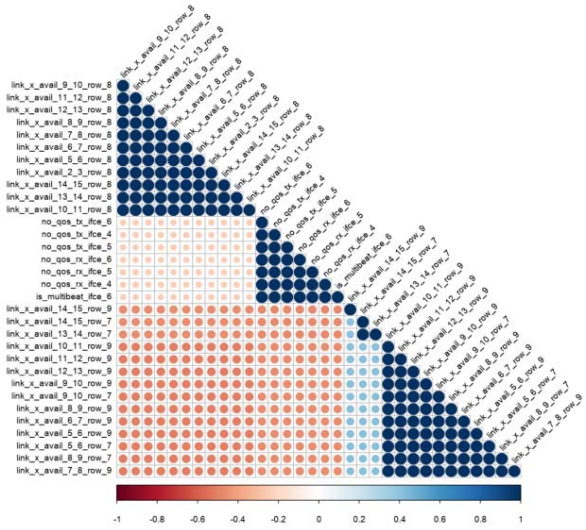

Fig. 2. The chart illustrates an entropy analysis between various feature dimensions. Entropy is one of several ways to measure diversity. An index value of 1, indicated in blue on the bottom axis, represents true diversity. A distance between features (measured by the index) is defined as a linear function of entropies of the features. This metric generally gives a good indication of feature correlation and information loss upon feature dimension reduction.

various feature dimensions. Entropy is one of several ways to measure diversity in the input data.

Different SoCs have different sizes, agent positions and hence different amounts of available NoC routing channels and ports for agents to connect. Using this information in absolute terms while creating feature sets is inefficient because it is unlikely to find many SoC designs with identical absolute features. Instead, we propose a method to normalize the topological features into a fixed size grid and use this normalized information in the predictor so as to find stronger correlations between the features of different SoC specifications. Fig. 3a illustrates an original grid with agent NoC ports and NoC routing blockages. Fig. 3b illustrates a grid obtained after ignoring agents and treating overlapping links as blockages. Fig. 3c illustrates a grid obtained after ignoring NoC routing blockages. Fig. 3d illustrates a grid after removing columns and rows at which no ports are connected. Fig. 3e illustrates columns height and rows width being adjusted. Fig. 3f illustrates a look of a 2x expanded grid. Note that the expanded grid is an identical structure of the original grid. A larger grid can be shrunk to a smaller grid using a similar process. This technique is used to resize SoCs of different sizes into a normalized size that is typically 4 or 16 horizontal rows and vertical columns each.

B. Strategies for NoC Design

Different interconnect strategies that define the design space are used to realize an interconnect design. Some examples of design strategies include the choice of NoC topology such as a mesh, ring, crossbar, tree, torus etc.; routing choices between agents such as using single turn versus multi-turn routes; different arbitration policies within

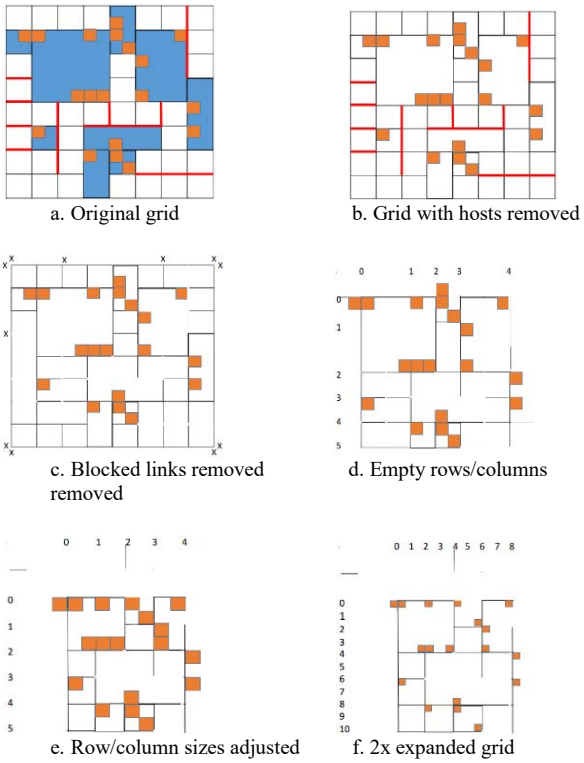

Fig. 3. Process of transforming a SoC specification into a normalized grid

the NoC such as round-robin, strict priority, dynamic priority; clock and power assignments within the NoC; sizing of buffers inside the NoC; use of multiple parallel NoCs (layers) versus a single NoC; different cost functions tailored to the requirements of the NoC such as latency, power, area and bandwidth; separation or aggregation of different types of messages within the NoC.

Multiple interconnect design strategies are used in conjunction with one another to form a strategy vector that is used to create a NoC design. It should be noted that strategies can overlap with and influence one another which increases the design space. For example, the number of layers used in the NoC design affects the number of choices for separation or aggregation of messages in the NoC and the routing choices. It is important to define the interconnect design space carefully and efficiently so that 1) several good designs are available in this space, 2) it remains feasible to develop prediction models with a reasonable amount of data and training in terms of compute and memory resources and 3) there is minimal conflict between different design strategy choices.

C. Quality Metrics

While every NoC has a fixed requirement of providing complete connectivity between agents while ensuring freedom from protocol-level and network-level deadlocks, the qualitative goals of the interconnect depend on the SoC specification (features). For the purpose of qualifying NoC designs, we propose the definition of different quality

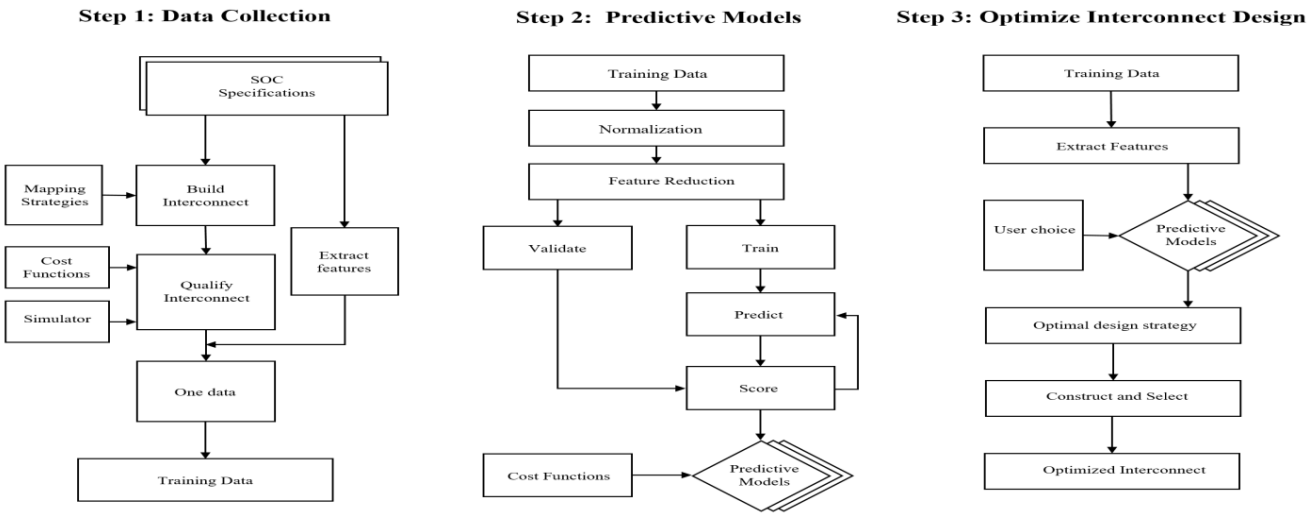

Fig. 4. The 3 Step process of using Machine Learning for NoC Design

metrics that use combinations of NoC bandwidth, packet latency, end-to-end latency, number of wires, wire density, number of flops, flop density, dynamic and leakage power based on the characteristics of the SoC and design goals of the NoC. In our experiments, we show that it is useful to build different predictive models for different design goals.

V. Training and Generation of Predictive Models

The MLNoC scheme uses a three step process illustrated in Fig. 4 that involves the collection of training data, the generation of predictive models and the querying of the model to obtain an optimized interconnect.

A. Training Models and Methods

As in most use cases of ML, applying it to NoC design involves the creation of a trained model that is subsequently used to predict an output for a given input. In the context of NoC design, the input to the predictor is the SoC features and the output is a set of design strategies that would yield an optimal interconnect for the specified SoC architecture. Large sets of training data are required in order to train and build a predictive model. We generate training data using thousands of SoC designs having a wide range of characteristics. For each SoC design, many sets of interconnect design strategies are explored and a quality metric is recorded for each of the constructed interconnects. It is important for there to be a high degree of variance in the types of SoC designs and the interconnect design strategies so as to train the model to work effectively for all types of designs.

B. Generating Predictive Models

The training data comprising of the set of input feature vector, design strategy sets and quality for each of them are used in the generation of predictive models. The model is built using standard algorithms such as random forest, neural network, multi-variant linear regression, vector machines, or pattern matching.

1) Random Forest Algorithm

The input dataset for training consists of the features for varying SoC designs. The random forest model is created by randomly selecting multiple subsets of the input dataset and building decision trees for each of the subsets.

2) Neural Networks

The input to the neural network is the feature set for various known SoC designs and the output of the neural network is the predicted set of strategies to be used for the NoC construction. For each known SoC design, there is also a known optimal NoC design strategy. The neural network is trained across these multiple SoC designs by computing the predicted set of strategies and back propagating based on the error between the known optimal strategy and predicted strategy. Once trained, based on any SoC design, the neural network predicts the sets of strategies to be used for NoC construction.

3) Multi-variant Linear Regression

Another approach is to use multi-variant linear regression. The dependent variables in the context of interconnect construction are the strategy vectors and the independent features are the various SoC features defined previously. Once trained, the model can be used to predict the set of strategies for optimal interconnect construction based on the SoC features via linear regression. The known SoC designs along with set of strategies that yield the best interconnect for each SoC is used as the training data to build the linear regression model.

C. Optimize New Designs

Once the training is completed and predictive models have been generated, new specification can be fed to the desired ML predictor (based on the metrics to be optimized). The predictor can quickly sort through and identify promising combinations of strategies that will most likely

TABLE I. DIFFERENT SOC DESIGN AND THEIR CHARACTERISTICS

Name	Type	Market Space	Achieved Bandwidth (GBps)	Memory Capacity (GBps)	Clock Frequency (GHz)	Low Power	SoC Size (mm^2)	NoC Size	Process Node (nm)	Number of agents
SoC1	Coherent	Desktop	32	204.8	1.6	No	88	3x2	28	6
SoC2	Coherent	FPGA	8	36	0.75	Yes	135	6x4	22	10
SoC3	Non-coherent	Graphics	14	8	0.125	No	36	3x3	28	6
SoC4	Non-coherent	IoT	6	16	0.5	Yes	18	3x2	n/a	7
SoC5	Coherent	Networking	128	32	0.5	No	104	8x4	14	34
SoC6	Coherent	Mobile	96	128	1	Yes	100.5	5x5	n/a	21
SoC7	Coherent	Networking	64	200	1	No	126	2x3	7	25
SoC8	Coherent	Data Center	144	256	1	No	338	5x4	14	10
SoC9	Coherent	Consumer	64	56	0.5	Yes	100	4x6	10	13
SoC10	Non-coherent	Mobile	996	1344	1	Yes	168	5x6	7	114
SoC11	Non-coherent	Mobile	324	2304	1	Yes	163	6x8	22	45
SoC12	Non-coherent	Graphics	48	76.8	0.6	No	114	4x4	14	13
SoC13	Non-coherent	Consumer	64	128	2	Yes	63	3x6	14	19
SoC14	Non-coherent	Consumer	32	135.168	0.65	Yes	256	5x4	n/a	22
SoC15	Non-coherent	Mobile	76	80.64	0.63	Yes	234	2x3	28	13
SoC16	Coherent	Automotive	16	68.224	2	No	120	4x3	n/a	4
SoC17	Coherent	Mobile	24	25.6	0.8	Yes	100	3x3	10	13
SoC18	Coherent	Consumer	48	102.4	0.8	No	98.5	4x7	10	18
SoC19	Non-coherent	Graphics	32	68.096	0.532	No	132	4x3	14	11
SoC20	Coherent	Design service	73	76.8	0.8	No	110	3x3	n/a	9
SoC21	Non-coherent	Design service	98	35.6	0.6	No	132	5x5	22	13
SoC22	Coherent	Consumer	48	48	0.5	Yes	125	4x4	7	14
SoC23	Coherent	Graphics	128	409.6	0.8	No	256	7x11	14	36
SoC24	Non-coherent	Consumer	16	19.6	0.5	No	89	6x4	n/a	14
SoC25	Non-coherent	Mobile	12	12.544	0.4	Yes	42	5x4	10	20
SoC26	Non-coherent	IoT	8	25.6	0.266	Yes	256	12x11	10	44
SoC27	Coherent	Automotive	196	204.8	2	No	208	5x4	14	30
SoC28	Coherent	Automotive	136	136.8	2.6	No	192	4x4	n/a	19
SoC29	Non-coherent	Automotive	256	288	0.5	No	116	3x4	22	19
SoC30	Coherent	Mobile	72	76.8	2	Yes	142	5x4	n/a	9

yield an optimal design based on the given metric. It is also possible to query the predictor for the quality of any given combination of input features and design strategies. This allows quick analysis of how well any particular combination of design strategies may work for a given SoC without having to design and analyze a full-blown interconnect.

VI. EXPERIMENTS AND RESULTS

In this section, we evaluate the performance of our MLNoC scheme compared to baseline NoC designs that have been created by seasoned SoC architects. We also evaluate the quality of the NoCs generated with increased training of the ML model over time.

A. SoC Workloads

We evaluate the performance of interconnects designed using our MLNoC scheme on 30 real world SoC designs spanning a wide variety of applications. The key features and market segments for all SoC designs are listed in Table 1. We include both coherent and non-coherent SoC designs from multiple market segments including data centers, desktops, networking devices, mobile chips, consumer hardware, FPGA and automobile. The SoC designs have widely varying characteristics with chip sizes varying from $16mm^2$ to $338mm^2$, clock frequencies varying from 0.125 GHz to 2.66 GHZ, number of agents varying from 4 to 114 and total bandwidth requirements ranging from 8 GBps to 996 GBps. Note that these SoC designs were not used during the training of the MLNoC predictor.

B. Comparision against Architected NoCs

In order to evaluate the quality of NoCs produced using MLNoC, for each SoC, we compare the interconnect designs against baseline interconnects that were designed by seasoned engineers and architects. For the purpose of NoC construction and performance analysis, we use a NoC compiler that uses the SoC specification as an input and generates a NoC using given a list of design strategies. The quality metrics are evaluated using a built-in cycle accurate model of the NoC.

Different SoC designs have different design goals such as performance, area, power or a combination of them based on different use cases. The below experiments illustrate that it is efficient to use different predictive models based on the design goal that needs to be optimized. In our experiments, we use 3 different predictive models – balanced, performance and area.

1) Balanced Model

The first model is tailored to a "balanced" goal with equal weightage to maximizing performance in terms of overall NoC bandwidth and minimizing the number of flops (area) in the NoC. MLNoCs achieve an average of about 45% higher performance than baseline NoCs with about 12% lesser area.

2) Performance Model

The second model is built with a "performance" goal that tends to favor maximizing the overall bandwidth of the NoC even if it comes at the cost of slightly higher area compared to a baseline NoC. MLNoCs achieve an average of about 54% higher performance than baseline NoCs with about 6% lower area.

3) Area Model

The third model is built with an "area" goal that tends to favor minimizing the area in the NoC even if it comes at the cost of slightly reduced system performance. MLNoCs achieve an average of about 15% lower with a 41% higher performance than baseline NoCs.

Fig. 5 represents the comparison of a standardized quality of an MLNoC from the three different models versus the baseline for each SoC workload respectively. It illustrates that while an MLNoC produced by any of the three predictive models always yields results with a higher quality than a baseline NoC, no one single model outperforms the rest for every design. Different models yield the best NoC for different designs.

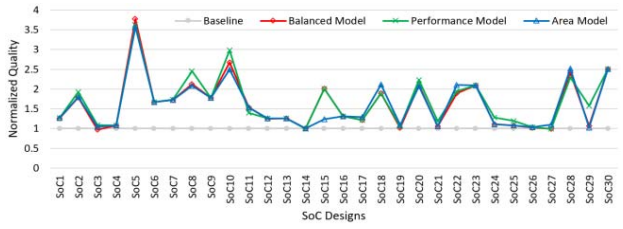

Fig. 5. Normalized quality across all SoC designs for balanced, performance and area models

Since different SoCs have different design goals, it is useful to choose a predictive model that is tuned to optimize the metrics that are most important to the design under consideration. Fig. 6 shows a comparison of the average increase in bandwidth achieved on designs in different market segments using three predictive models. It can be seen that the Performance model that has been tuned to maximize the bandwidth always yields a greater improvement in performance with respect to the baseline compared than the Balanced and Area models.

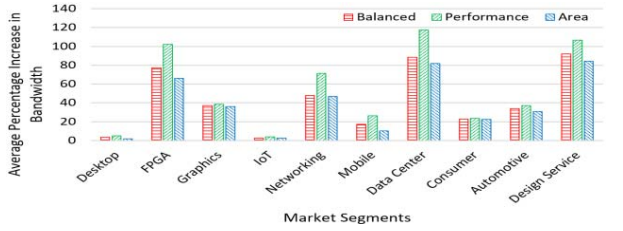

Fig. 6. Comparison of average percentage improvement in bandwidth using balanced, performance and area models

Fig. 7 shows a comparison of the average reduction in number flops achieved on designs in different market segments using the three predictive models. The Area model that has been tuned to minimize the area always yields a higher reduction in area with respect to the baseline compared to the Balanced and Performance models. It should be noted that although the average number of flops in MLNoCs produced by the Balanced or Performance models in SoC designs in the FPGA market segment is be worse than the baseline case, this degradation in area is accompanied by a substantially higher increase in bandwidth that offsets the increase in area and hence improves the quality of the NoC. Similarly, a reduction in bandwidth in MLNoCs produced using the Balanced or Area models is always accompanied by a reduction in the number of flops that is far greater and offsets the degradation in bandwidth, thus improving the overall quality of the NoC.

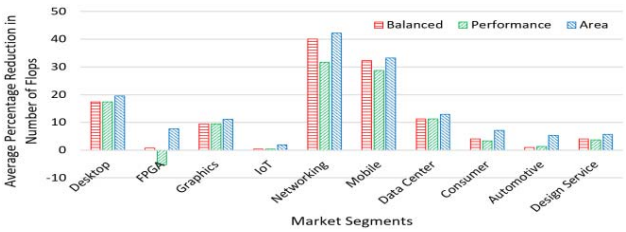

Fig. 7. Comparison of average percentage improvement (reduction) in number of flops using balanced, performance and area models

C. *Effect of Training on MLNoC*

The results predicted from a ML model can only be as good as the selection of features and strategies for NoC construction and the breadth and diversity of the training data provided to it. It is impossible for MLNoC to predict strategies that have not been explored as part of the training of the model or to correctly process SoC characteristics that are alien to it.

We now demonstrate the progression in the quality of results from MLNoC over different versions of the model. Starting from the first version – v1, we periodically increase the training set for the model and also add new features and strategies that can increase the chances of finding the best NoC design for a SoC. Theoretically, the increase in training data will result in higher quality of NoCs as we progress to newer versions due to the increased probability of finding a solution with a higher quality.

As evidenced in the Fig. 8, the average quality of results across all SoC designs progressively increases to about 150% more than the baseline as we move to newer versions of MLNoC. There is a case of the quality of results remaining the same across versions 5 and 6. This is due to the addition of features and strategies that increase the dimensions and search space without improving the design space. Thus, it is critically important to carefully select SoC features and strategies for the NoC design and ensure diversity in the training data.

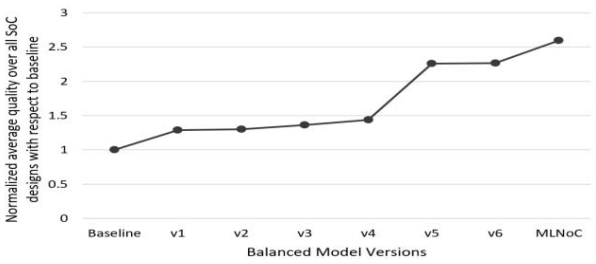

Fig. 8. Normalized quality across all SoC designs for different versions of a balanced predictive model

VII. CONCLUSION

This paper introduces MLNoC, a novel method to address the growing challenges associated designing optimal NoCs for complex SoCs by employing supervised machine learning. MLNoC utilizes machine learning to augment an automated NoC design flow so that optimal NoC designs can be obtained quickly and reliably for a wide variety of requirements and market segments. A collection of pre-trained models are employed each of which are geared towards predicting design approaches that are most likely to optimize certain quality metric. The models are built for carefully chosen SoC features that are likely influence the NoC design the most and design approaches that yield maximum coverage of the design space. We demonstrate that MLNoC consistently builds NoCs with 40-54% higher performance and 6-15% lower area compared to designs hand tuned and optimized. The phenomena continues for a wide variety of user defined quality metrics providing users an unprecedented opportunity and flexibility to achieving various design goals. MLNoC represents a significant advancement in automatic SoC interconnect design and analysis and opens up new possibilities for advanced system architecture and design.

REFERENCES

[1] W.J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference*, 2001.

[2] S. Kumar, A. Jantsch, J.P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja and A. Hemani, "A network on chip architecture and design methodology," in *VLSI*, 2002.

[3] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," in *Computer*, Volume 35, Issue 1, January 2002.

[4] T. Simunic and S. Boyd, "Managing power consumption in networks on chips," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, 2002..

[5] A.N. Udipi, N. Muralimanohar and R. Balasubramonian, "Towards scalable, energy-efficient, bus-based on-chip networks," in *HPCA*, 2010.

[6] B. Yang, T.C. Xu, T. Santti and J. Plosila, "Tree-model based mapping for energy-efficient and low-latency Network-on-Chip," in *DDECS*, 2010.

[7] T. Kam Ho, "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition*, 1995.

[8] T. Kam Ho, "The random subspace method for constructing decision forests," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 20, Issue 8, August 1998.

[9] T. Kam Ho, J.J. Hull and S.N. Srihari, "Decision combination in multiple classifier systems," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 16, Issue 1, January 1994.

[10] L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U.A. Muller, E. Sackinger, P. Simard, V. Vapnik et al, "Comparison of classifier methods: a case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International. Conference on Pattern Recognition*, 1994.

[11] Y.S. Huang and C.Y. Suen, "A method of combining multiple classifiers-a neural network approach," in *Proceedings of the 12th IAPR International. Conference on Pattern Recognition*, 1994.

[12] V. Uebele, S. Abe and M. Lan, "A neural-network-based fuzzy classifier," in *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 25, Issue 2, Feb 1995.

[13] S. Mitra and S.K. Pal, "Self-organizing neural network as a fuzzy classifier," in *IEEE Transactions on Systems, Man, and Cybernetics*, Volume 24, Issue 23, Feb 1995.

[14] W. Pedrycz, "Fuzzy neural networks with reference neurons as pattern classifiers," in *IEEE Transactions on Neural Networks*, Volume 3, Issue 5, Sep 1992.

[15] Y.J. Yoon, P. Mantovani and L.P. Carloni, "System-level design of networks-on-chip for heterogeneous systems-on-chip," in *NOCS*, 2017.

[16] J. Sun and Y. Zhang, "An energy-aware mapping algorithm for mesh-based network-on-chip architectures," in *PIC*, 2017.

[17] H.C. Touati and F. Boutekkouk, "A weighted minimal fully adaptive congestion aware routing algorithm for Network on Chip," in *EDiS*, 2017.

[18] G.S. Anirudh and Soumya J, "Routing Algorithm for Application-Specific Network-on-Chip with Irregular Core Sizes," in *iNIS*, 2017.

[19] J. Khichar, S. Choudhary and R. Mahar, "Fault tolerant dynamic XY-YX routing algorithm for network on-chip architecture," in *I2C2*, 2017.

[20] J. Dai, X. Jiang, R. Li and T Watanabe, "An Efficient Deadlock-Free Adaptive Routing Algorithm for 3D Network-on-Chips," in *MCSoC*, 2017.

[21] S. Sariga and C. Nandagopal, "An area efficient network on chip architecture using high performance pipelines FIFO technique," in *ICEICE*, 2017.

[22] C. Rajamanikkam, JS. Rajesh, K. Chakraborty and S. Roy, "BoostNoC: Power efficient network-on-chip architecture for near threshold computing," in *ICCAD*, 2016.

[23] S. Sahu and H.M. Kittur, "Area and power efficient network on chip router architecture," in *ICT*, 2013.

[24] M.A. Abd El ghany, M.A. El-Moursy, D. Korzec and M. Ismail, "Power efficient Networks on Chip," in *ICECS*, 2009.

[25] E. Offori-Attah and M.O. Agyeman, "A survey of recent contributions on low power NoC architectures," in *Computing Conference*, 2017.

[26] G. Leary, K.S. Chatha, "Automated technique for design of NoC with minimal communication latency," in *Proceedings of the 7th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS, 2009.*

[27] S.O. Bykov, "Algorithm for automated custom network-on-chip topologies design," in *East-West Design & Test Symposium*, 2013.

[28] B. Grot, S. W. Keckler, and O. Mutlu, "Preemptive virtual clock: a flexible, efficient, and cost-effective QoS scheme for networks-on-chip," in *MICRO*, 2009.